UMassAmherst

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model

Liu Yang[1], Qingyao Ai[1], Jiafeng Guo[2], W. Bruce Croft[1]

[1]Center for Intelligent Information Retrieval, University of Massachusetts Amherst

[2]Institute of Computing Technology, Chinese Academy of Sciences

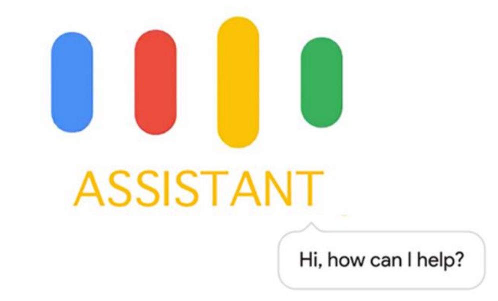
UMASS AMHERST

Outline

- **Motivation**
- **Related Works**
  - **Learning to rank for QA**
  - **Deep learning for QA**
- Attention-based Neural Matching Model
- Experiments
  - Data Set and Experiment Settings
  - Model Learning Results
  - Experimental Results for Ranking Answers
- Conclusions and Future Work

Motivation

- ***Question answering*** plays a central role in many popular mobile search systems and intelligent assistant systems
  - Google Assistant, Microsoft Cortana, Microsoft Xiaoice, IBM Watson, etc.
- Users are more likely to expect direct answers instead of a rank list of documents from search results
  - Retrieve finer grained text units such as ***passages or sentences*** as ***answers*** for ***Web queries*** or ***questions***



UMASS AMHERST

Learning to Rank for QA

- Many previous QA systems used a **learning to rank** approach
  - Encode question/answers with complex linguistic features including lexical, syntactic and semantic features
  - E.g. Surdeanu et al. [1,2] investigated a wide range of feature types for learning to rank answers
- Problems with learning to rank approaches
  - Reply on **feature engineering**, which is time consuming and requires domain dependent expertise
  - Need **additional NLP parsers** or **external knowledge sources**
    - may not be available for some languages

[1] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online QA collections. In ACL 2008.
[2] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. Comput. Linguist., 2011.

UMASS AMHERST

Deep Learning for QA

- Recently researchers have been studying **deep learning** approaches to learn semantic match between questions and answers
  - Convolutional Neural Networks (CNN) [3, 4, 5]
  - Long Short-Term Memory (LSTM) networks [6]
  - Benefit of not requiring hand-crafted linguistic features and external resources except pre-trained word embedding
  - Some of them [5] achieve state-of-the-art performance for ***answer sentence selection*** task benchmarked by the ***TREC QA Data***

[3] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep Learning for Answer Sentence Selection. In NIPS Deep Learning Workshop, 2014.
[4] X. Qiu and X. Huang. Convolutional neural tensor network architecture for community-based question answering.In IJCAI 2015.
[5] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In SIGIR 2015.
[6] D. Wang and E. Nyberg. A long short-term memory model for answer sentence selection in question answering. In ACL 2015.

Deep Learning for QA

- Problems with current deep learning architectures for answer sentence selection
  - The proposed models, either based on CNN or LSTM, need to be **combined with additional features** such as word overlap features [3,5] and BM25 [6] to perform well
  - Without combining additional features, the performance of their model is ***significantly worse***
    - Comparing with the results from the state-of-the-art methods using linguistic feature engineering [7]
- Research question:
  - Could we build deep learning models that can **achieve comparable or even better performance without combining additional features** than methods using feature engineering ?

[7] W.-t. Yih, M.-W. Chang, C. Meek, and A. Pastusiak. Question answering using enhanced lexical semantic models. In ACL 2013

Observations From the Current Deep Learning Architectures for Ranking Answers

- Architectures not specifically designed for question/answer matching
  - CNN
    - Uses position-shared weights with local perceptive filters to learn spatial regularities as in many CV tasks
    - Such spatial regularities may not exist in the semantic matching between questions and answers
    - Complex linguistic property of natural languages
  - LSTM
    - View the question/answer matching problem in a sequential way
    - No direct interactions between question and answer terms
    - Can not capture sufficiently detailed matching signals
- Our solution
  - Introduce a novel ***value-shared weighting scheme*** in deep neural networks
  - Learn ***value regularities*** rather than spatial regularities

Observations From the Current Deep Learning Architectures for Ranking Answers

- Lack of modeling question focus
  - Understanding the focus of questions which are important terms is helpful for ranking answers correctly
    - E.g. Where was the *first burger king* restaurant *opened* ?
  - Most existing text matching deep learning models do not explicitly model question focus
- Our solution
  - Incorporate ***attention scheme*** over question terms
    - Introduce attention mechanisms with a gating function
    - Explicitly discriminate the question term importance

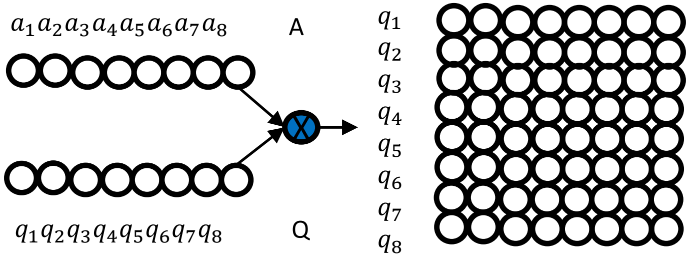
Outline

- Motivation
- Related Works
  - Learning to rank for QA
  - Deep learning for QA
- **Attention-based Neural Matching Model**
- Experiments
  - Data Set and Experiment Settings
  - Model Learning Results
  - Experimental Results for Ranking Answers
- Conclusions and Future Work

UMASS AMHERST

QA Matching Matrix

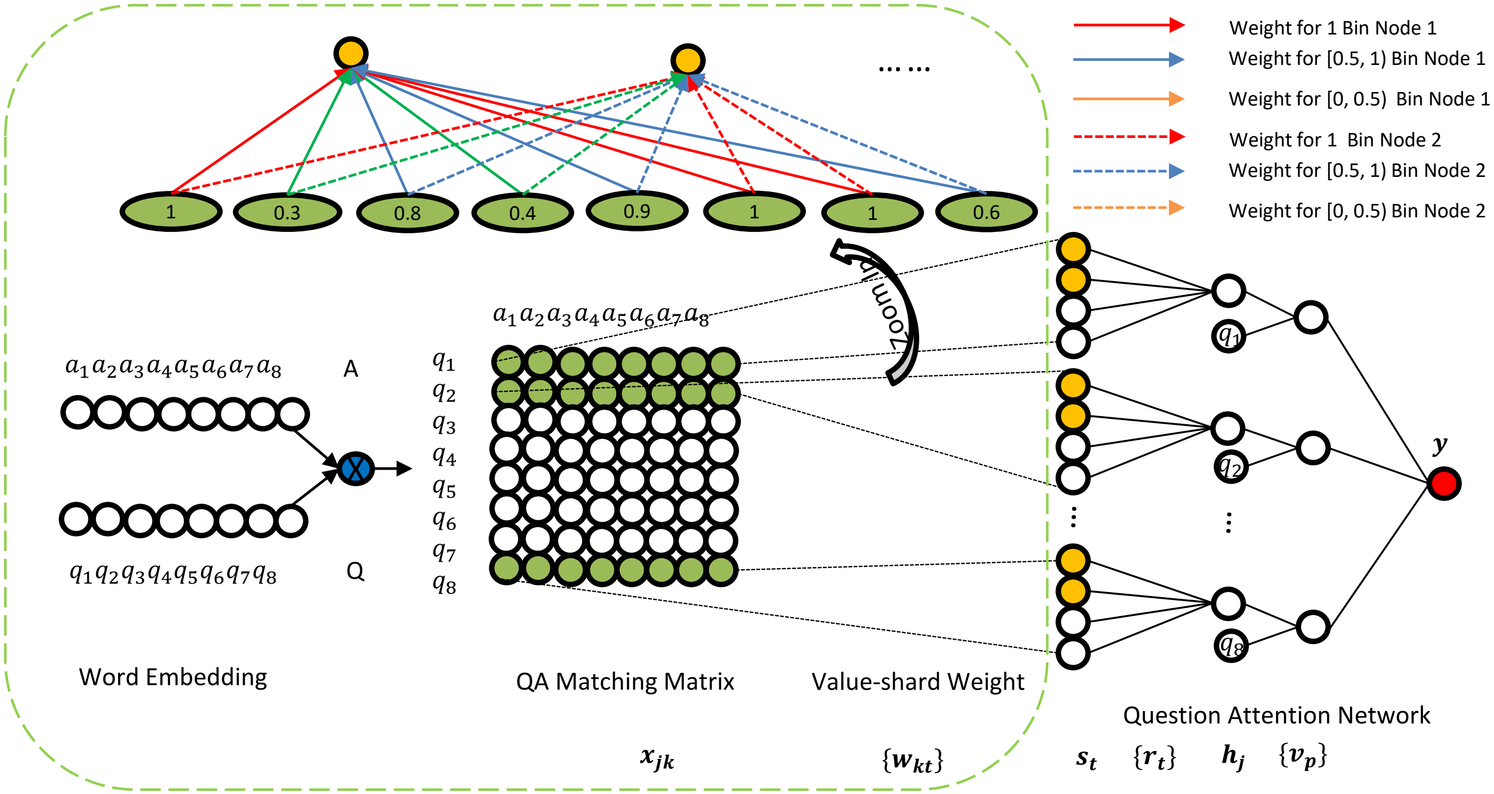
- QA Matching Matrix
  - A matrix represents the semantic matching information of term pairs from a question and answer pair
  - Given a question $\mathbf{q}$ with length M and an answer $\mathbf{a}$ with length N
    - An M by N matrix $\mathbf{P}$
    - $\mathbf{P}_{\mathrm{j,i}}$ is the sematic similarity between $\mathbf{q}_j$ and $\mathbf{a}_i$ using word embedding
    - Assign value 1 if $\mathbf{q}_j$ and $\mathbf{a}_i$ are the same term
    - Inspired by the ARC-II model proposed by Hu et al. [8]



[8] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In NIPS 2014.

UMASS AMHERST

Attention-based Neural Matching Model

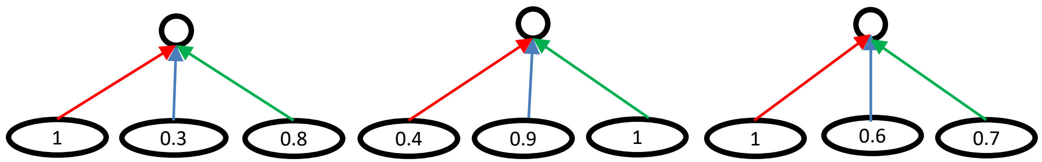


$$y = \sum_{j=1}^{M} \tau(\mathbf{v} \cdot \mathbf{q}_j) \cdot \delta\left(\sum_{t=0}^{T} r_t \boxed{\delta\left(\sum_{k=0}^{K} w_{kt} x_{jk}\right)}\right)$$

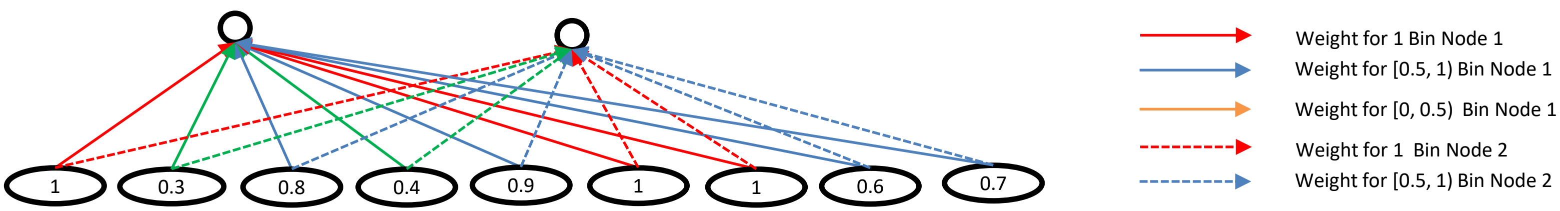
τ: softmax gate function
δ: sigmoid function

- Neural network architecture with **value-shared weights**

Value-shared Weighting



Position-shared Weight (CNN)



Value-shared Weight (aNMM)

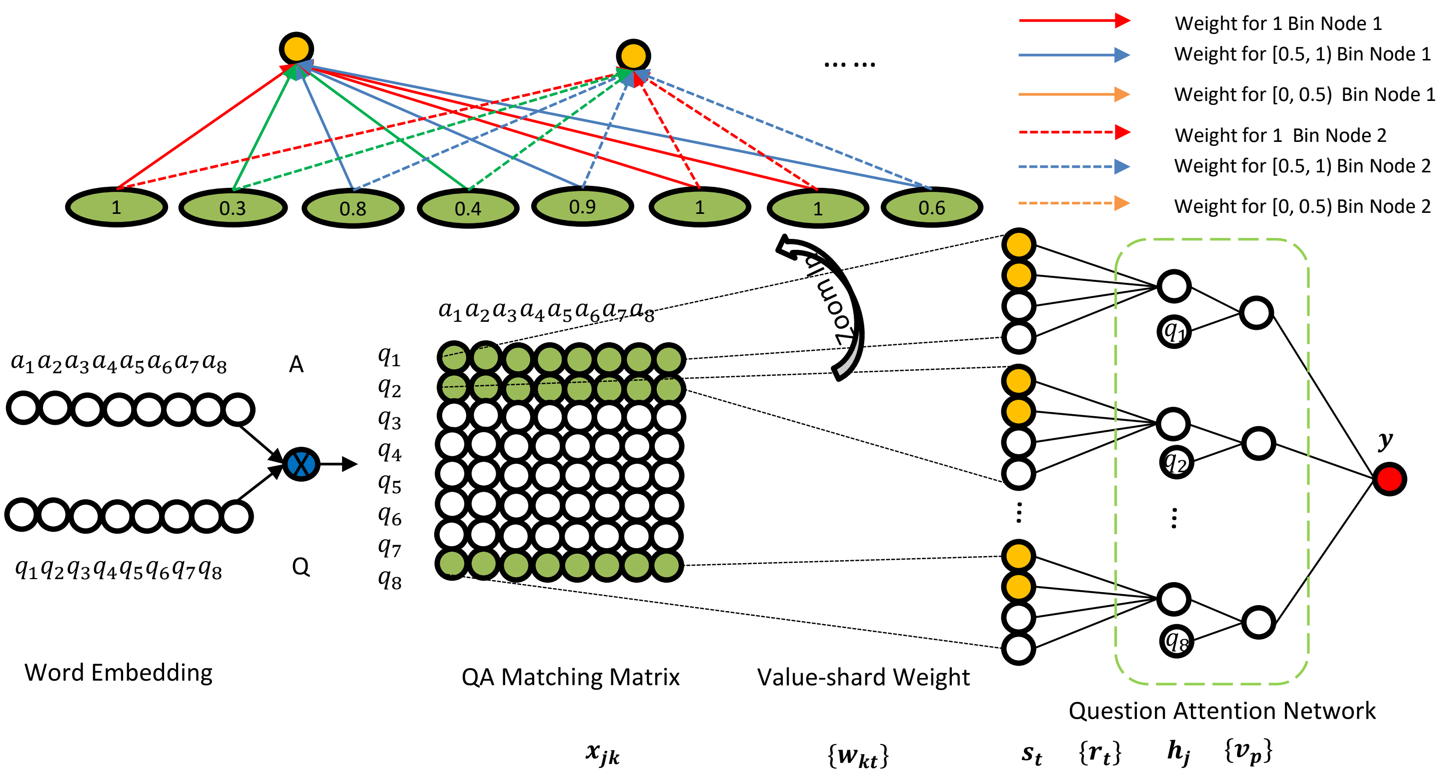
- In CNN, the weight associated with a node only depends on its **position** as specified by the filters
- In aNMM, the weight associated with a node depends on its **value**

$$y = \sum_{j=1}^{M} \tau(\mathbf{v} \cdot \mathbf{q}_j) \cdot \delta\left(\sum_{t=0}^{T} r_t \boxed{\delta\left(\sum_{k=0}^{K} w_{kt} x_{jk}\right))}$$

τ: softmax gate function
δ: sigmoid function

- Neural network architecture with **value-shared weights**

Question Attention Network

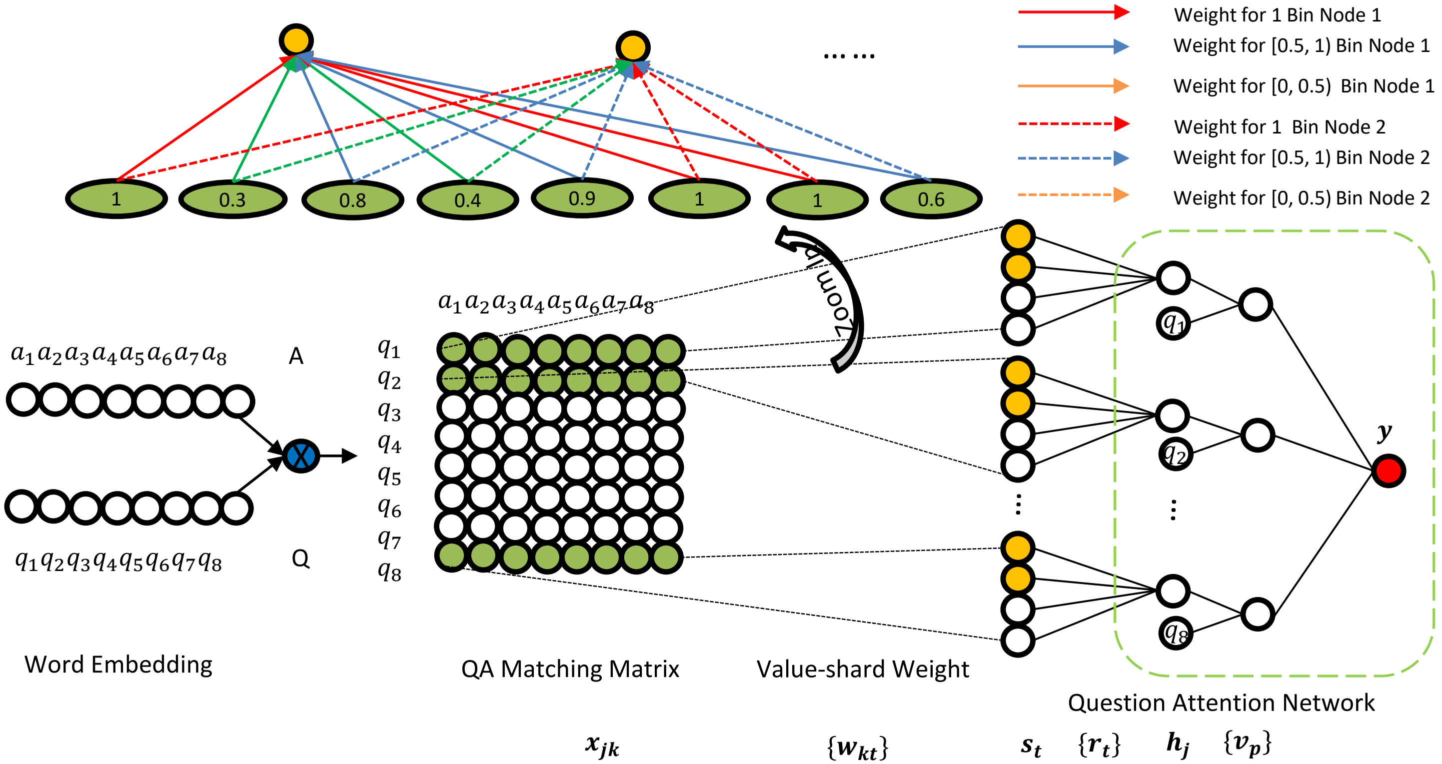


$$y = \boxed{\sum_{j=1}^{M} \tau(\mathbf{v} \cdot \mathbf{q}_j)} \cdot \delta\left(\sum_{t=0}^{T} r_t \cdot \delta\left(\sum_{k=0}^{K} w_{kt} x_{jk}\right)\right)$$

τ: softmax gate function
δ: sigmoid function

- Neural network architecture with **attention schemes**

Two Variations: aNMM-1 and aNMM-2



aNMM-1: $y = \sum_{j=1}^{M} \tau(\mathbf{v} \cdot \mathbf{q}_j) \cdot \delta\left(\sum_{k=0}^{K} w_k x_{jk}\right)$

aNMM-2: $y = \sum_{j=1}^{M} \tau(\mathbf{v} \cdot \mathbf{q}_j) \cdot \delta\left(\sum_{t=0}^{T} r_t \cdot \delta\left(\sum_{k=0}^{K} w_{kt} x_{jk}\right)\right)$

- Two variations
- aNMM-1: **basic architecture**
- aNMM-2: **Extension with multiple sets of value-share weights**

Back Propagation for Model Training

- Backward propagation with stochastic gradient descent
- Pairwise Learning
- Given a triple $(\mathbf{q}, \mathbf{a}^+, \mathbf{a}^-)$ where
  - $\mathbf{q}$ question sentence
  - $\mathbf{a}^+$ correct answer sentence
  - $\mathbf{a}^-$ wrong answer sentence
  - Hinge Loss function $e(\mathbf{q}, \mathbf{a}^+, \mathbf{a}^-; \mathbf{w}, \mathbf{r}, \mathbf{v}) = \max(0, 1 - S(\mathbf{q}, \mathbf{a}^+) + S(\mathbf{q}, \mathbf{a}^-))$
  - Compute $\Delta S = 1 - S(\mathbf{q}, \mathbf{a}^+) + S(\mathbf{q}, \mathbf{a}^-)$
  - If $\Delta S \leq 0$ Skip this triple
    - If $\Delta S > 0$ Compute the gradients w.r.t $\mathbf{v}, \mathbf{r}, \mathbf{w}$
    - Update the model parameters to minimize the loss function with BP algorithm

Outline

- Motivation
- Related Works
  - Learning to rank for QA
  - Deep learning for QA
- Attention-based Neural Matching Model
- **Experiments**
  - **Data Set and Experiment Settings**
  - **Model Learning Results**
  - **Experimental Results for Ranking Answers**
- Conclusions and Future Work

UMASS AMHERST

Experimental Data and Settings

- TREC QA data set from TREC QA track 8-13
  - One of the most widely used benchmarks for answer sentence selection/ranking
  - Contains a set of factoid questions with candidate answers which are limited to a single sentence
  - Judgements in TRAIN and TRAIN-ALL
  - Word embedding: pre-trained with English Wikipedia dump with the Word2Vec tool by Mikolov et. al [9, 10]
- Statistics of the TREC QA data set

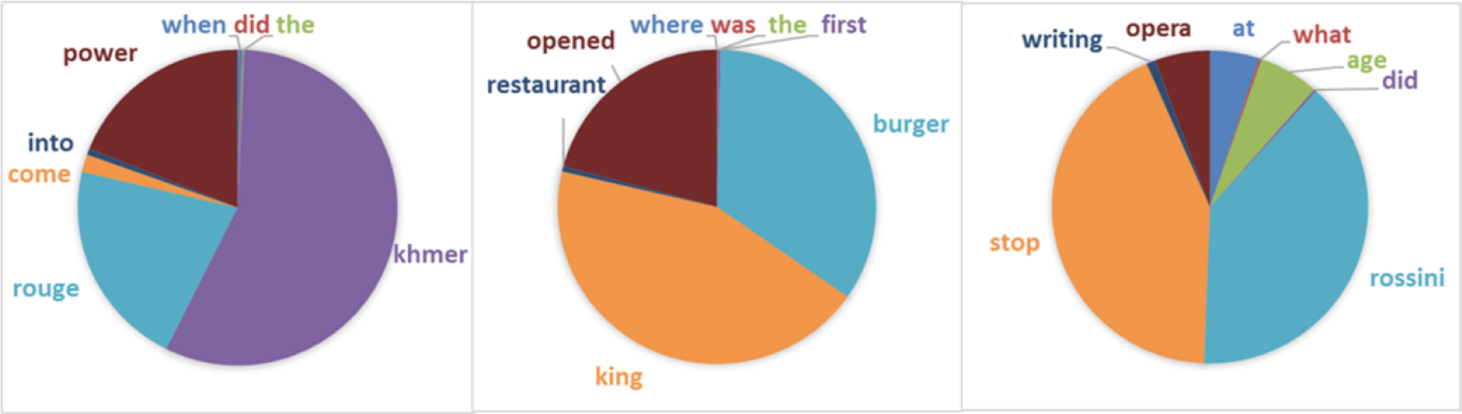
Data	#Questions	#QA pairs	%Correct	#Answers/Q	Judgement
TRAIN-ALL	1,229	53,417	12.00%	43.46	automatic
TRAIN	94	4,718	7.40%	50.19	manual
DEV	82	1,148	19.30%	14.00	manual
TEST	100	1,517	18.70%	15.17	manual

[9] https://code.google.com/archive/p/word2vec/
[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In NIPS 2013.

Model Learning Results

- Visualization of learned question term importance

test_14 Term Importance	when 4.91E-03	did 7.18E-04	the 8.97E-04	khmer 5.67E-01	rouge 2.13E-01	come 1.81E-02	into 6.59E-03	power 1.89E-01
test_66 Term Importance	where 2.16E-04	was 5.67E-04	the 1.96E-04	first 2.57E-03	burger 3.43E-01	king 4.39E-01	restaurant 5.35E-03	opened 2.08E-01
train_84 Term Importance	at 5.06E-02	what 2.54E-03	age 6.17E-02	did 2.68E-03	rossini 3.89E-01	stop 4.28E-01	writing 9.29E-03	opera 5.64E-02



Experimental Results

- Learning without combining additional features

Compare with methods using feature engineering (on TRAIN-ALL)

Method	MAP	MRR
Wang et al. (2007) [27]	0.6029	0.6852
Heilman and Smith (2010) [5]	0.6091	0.6917
Wang and Manning (2010) [26]	0.5951	0.6951
Yao et al. (2013) [31]	0.6307	0.7477
Severyn et al. (2013) [17]	0.6781	0.7358
Yih et al. (2013) [32]	0.7092	0.7700
aNMM-2	**0.7407**	0.7969
aNMM-1	0.7385	**0.7995**

Compare with deep learning methods

Training Data	TRAIN		TRAIN-ALL	
Method	MAP	MRR	MAP	MRR
Yu et al. (2014) [34]	0.5476	0.6437	0.5693	0.6613
Wang et al.(2015) [25]	/	/	0.5928	0.6721
Severyn et al. (2015) [18]	0.6258	0.6591	0.6709	0.7280
aNMM-2	0.7191	0.7974	**0.7407**	0.7969
aNMM-1	**0.7334**	**0.8020**	0.7385	**0.7995**

- Achieve better performance comparing with other methods using feature engineering
- Show significant improvements comparing with previous deep learning methods
- Results of aNMM-1 and aNMM-2 are very close
- aNMM-1 could be trained with higher efficiency

Experimental Results

- Learning with combining additional features

Compare with deep learning methods
Severyn et al. (SIGIR 2015) is the state-of-the-art result

Training Data	TRAIN		TRAIN-ALL	
Method	MAP	MRR	MAP	MRR
Yu et al. (2014) [34]	0.7058	0.7800	0.7113	0.7846
Wang et al. (2015) [25]	/	/	0.7134	0.7913
Severyn et al. (2015) [18]	0.7329	0.7962	0.7459	0.8078
aNMM-2	0.7306	0.7968	0.7484	0.8013
aNMM-1	**0.7417**	**0.8102**	**0.7495**	**0.8109**

Overview of previously published results on TREC QA data (the best setting of each model trained on TRAIN-ALL)

Method	MAP	MRR
Wang et al. (2007) [27]	0.6029	0.6852
Heilman and Smith (2010) [5]	0.6091	0.6917
Wang and Manning (2010) [26]	0.5951	0.6951
Yao et al. (2013) [31]	0.6307	0.7477
Severyn et al. (2013) [17]	0.6781	0.7358
Yih et al. (2013) [32]	0.7092	0.7700
Yu et al. (2014) [34]	0.7113	0.7846
Wang et al. (2015) [25]	0.7134	0.7913
Severyn et al. (2015) [18]	0.7459	0.8078
aNMM	**0.7495**	**0.8109**

- Combine the score of aNMM-1/aNMM-2 with QL score
- With the combined feature, both aNMM-1 and aNMM-2 have better performances
- aNMM-1 also outperforms CDNN by Severyn et al. ([5] in SIGIR 2015) which is the current state-of-the-art method

Outline

- Motivation
- Related Works
  - Learning to rank for QA
  - Deep learning for QA
- Attention-based Neural Matching Model
- Experiments
  - Data Set and Experiment Settings
  - Model Learning Results
  - Experimental Results for Ranking Answers
- **Conclusions and Future Work**

Conclusions and Future Work

- Propose an attention based neural matching model for ranking short answer text
  - Adopt value-shared weighting scheme instead of position-shared weighting scheme for combining matching signals
  - Incorporate question term importance learning using a question attention network
- Perform a thorough experimental study with TREC QA data and show promising results
  - Without combining additional features
    - Outperform previous deep learning methods and feature engineering methods with large gains
  - With one simple additional feature
    - Outperform the state-of-the-art method

Conclusions and Future Work

- Additional results on Microsoft Research WikiQA data [11]
  - Double confirms the advantages of the attention based neural matching models for ranking answer sentences.

Method	MAP	MRR
WordCount	0.4891	0.4924
WeightedWordCount	0.5099	0.5132
LCLR	0.5993	0.6086
PV	0.5110	0.5160
CNN	0.6190	0.6281
PV-Count	0.5976	0.6058
CNN-Count	0.6520	0.6652
aNMM-2	0.6455	0.6527
aNMM-1	**0.6562**	**0.6687**

- Future work
  - Extend our work to include non-factoid question answering data sets
    - Yahoo CQA /Stack Overflow/ WebAP
  - Interactive QA & Natural language dialogue for FAQ search

[11] Yi Yang, Wen-tau Yih and Christopher Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering, In EMNLP'15.

Thank You

Q&A

Email: lyang@cs.umass.edu
https://sites.google.com/site/lyangwww/